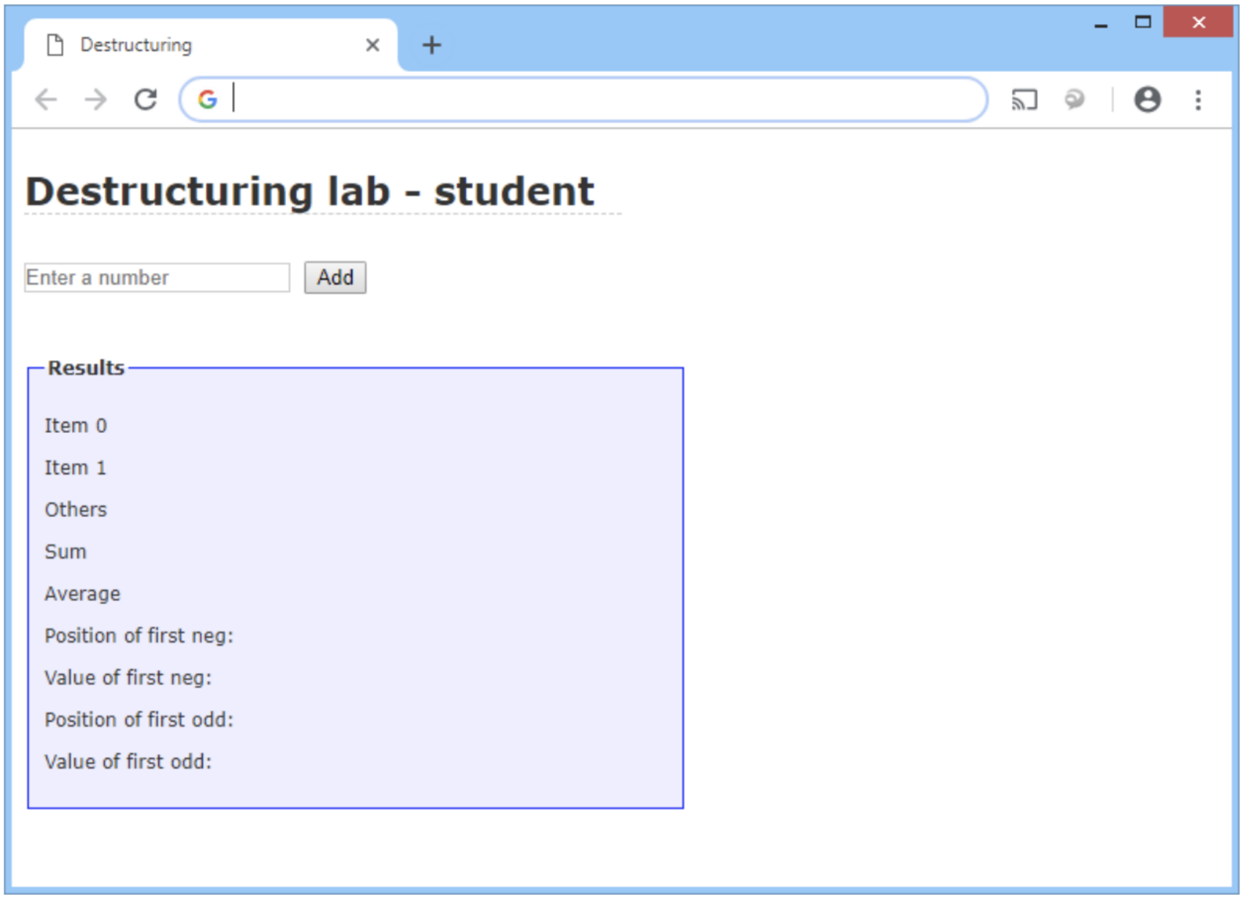
Labs

There are 6 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Getting started
2. Destructuring an array
3. Destructuring an object
4. (Extra credit 1) **More destructuring an object**
5. (Extra credit 2) **Swapping some variables**
6. (Extra credit 3) **Configuration object**

1. Getting started

Open up the student folder for this lab. It contains an HTML page and CSS stylesheet. Open the HTML page in a browser - it allows the user to enter a series of numbers (one at a time) and displays various details about the numbers in a results panel.



The web page doesn't do anything at the moment - you'll need to add some script to make it work. You'll write your code in `es6scripts/script.js` and then use Babel to transpile it into `es5scripts/script.js` (the HTML page picks up the ES5 version of the code).

To get the Babel transpiler running as a Gulp taks, open up a terminal and run `npx gulp` in the project directory.

2. Destructuring an array

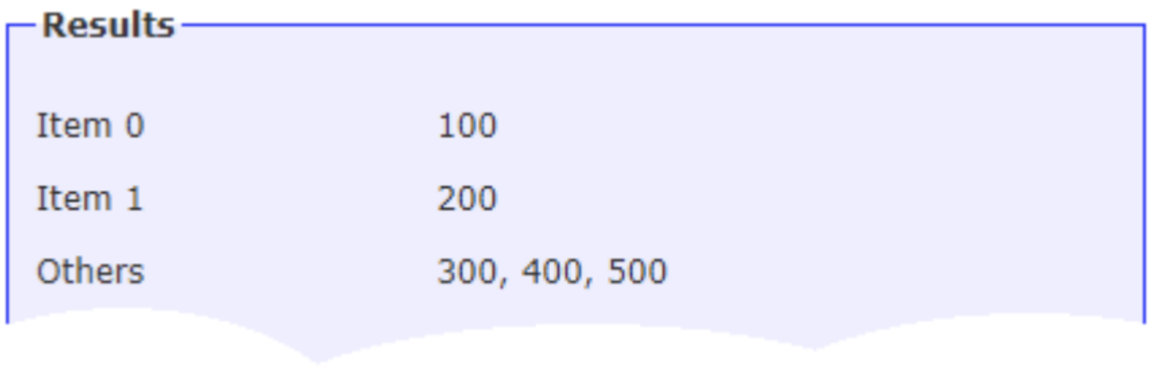
Take a look at the code in es6scripts/script.js. We've already written some starter code to handle clicks to the Add button. The code gets the number entered by the user, converts it into an integer, and adds it to a global array named numbers.

Where indicated by the "Exercise 2" comment, add code as follows:

- Destructure numbers into three variables holding item 0, item 1, and all the remaining elements from the array.

- Display these three variables on the HTML page, in the elements named item0, item1, and itemOthers. We've provided a function named setHtml() to make it easier to display some HTML content in an element; the 1st parameter is a CSS selector for an element, and the 2nd parameter is the HTML you want to display.

Save your file, and verify Babel transpiles it successfully. Then refresh the HTML page in the
browser and add some numbers. Verify the results are displayed correctly. For example, if you entered numbers such as 100, 200, 300, 400, 500 you should see something like this:



Item 0	100
Item 1	200
Others	300, 400, 500

3. Destructuring an object

Write a function named `stats()` that returns an object containing 2 properties:

- **sum** – The sum of all the values in the numbers array
- **average** – The average of all the values in the numbers array

Call `stats()` from the `doAdd()` function, where indicated by the "Exercise 3" comment. Destructure the returned object into two separate variables, holding the sum and average respectively. Display these values in the elements named sum and average.

Save your file, and verify Babel transpiles it successfully. Then refresh the HTML page in the
browser and add some numbers. Verify the sum and average values are updated every time you
add another number.

4. (Extra credit 1) More destructuring an object

Write a function named `find()` that finds the first element in the numbers array that matches
a specified test. Here are some hints:

- `find()` should take a "predicate" function as a parameter, which specifies the test to perform upon each element in the array.
- `find()` should iterate through the numbers array and invoke the predicate function upon each element until the predicate function returns true. At that point, return an object containing the index of the matching element, along with its value.
- If no match is found, return an object with suitable "error" properties.

Call `find()` from the `doAdd()` function, where indicated by the "Exercise 4" comment. Pass an arrow function as a parameter, to test if a value is negative. The find() function will return an object indicating the position of the first negative element, plus its value. Destructure this object and display its constituent fields in the elements named negPosition and
negValue.

Call `find()` again, this time passing in an arrow function that tests if a value is odd. The `find()` function will return an object indicating the position of the first odd element, plus its value. Destructure this object and display its constituent fields in the elements named oddPosition and oddValue.

Save your file, and verify Babel transpiles it successfully. Then refresh the HTML page in the browser and add some numbers. Verify the Web page displays the correct info about the first negative value and the first odd value.

5. (Extra credit 2) Swapping some variables

Swap two variables using on destructuring assignment.

```
let text1 = 'swap';
let text2 = 'me';

//Write Code here
```

6. (Extra credit 3) Configuration object

Suppose the following configuration object of a financial chart is given:

```
let config = {
    chartType : 0,
    bullColor : 'green',
    bearColor : 'red',
    days      : 30
};
```

Complete the function signature below such that the function may be called with any `config` objects (`null` and `undefined` are not allowed as inputs). If any of the four keys are missing, substitute their default values. The default values are the same as in the example configuration object.

```
function drawChart( data, /* Write your code here */ ) {
  // do not implement chart drawing functionality or anything
  // return {chartType, bullColor, bearColor, days};
};
```

```
const numbers = []

document.addEventListener('DOMContentLoaded', () => {
    document.getElementById('add').addEventListener('click', doAdd)
})

function doAdd() {

  const numberElem = document.getElementById('number')
    const number = parseInt(numberElem.value)
    numbers.push(number)
    numberElem.value = ''

    // Exercise 2 - Array destructuring.
    const [item0, item1, ...itemOthers] = numbers
    setHtml('#item0', item0)
    setHtml('#item1', item1)
    setHtml('#itemOthers', itemOthers.join(', '))

    // Exercise 3 - Object destructuring.
    const {sum, average} = stats()
    setHtml('#sum', sum)
    setHtml('#average', average)
    // Exercise 4 - More object destructuring.
    const {position: negPosition, value: negValue = 'n/a'} = find(n => n < 0)
    setHtml('#negPosition', negPosition)
    setHtml('#negValue', negValue)

    const {position: oddPosition, value: oddValue = 'n/a'} = find(n => n % 2 != 0)
    setHtml('#oddPosition', oddPosition)
    setHtml('#oddValue', oddValue)
}

function setHtml(selector, html) {
  const element = document.querySelector(selector)
  if (element) {
    element.innerHTML = html
  }
}

function stats() {
    let sum = 0
    for (let i in numbers) {
        sum += numbers[i]
    }
    return {sum: sum, average: sum/numbers.length}
}

function find(predicate) {
    let sum = 0
    for (let i in numbers) {
        let num = numbers[i]
        if (predicate(num)) {
            return {position: i, value: num}
        }
    }
    return {position: -1}
}
```